4.7 MULTICAST DATA DISTRIBUTION ON THE AWIPS LOCAL AREA NETWORK

Michael R. Biere*[?]
Darien L. Davis
Forecast Systems Laboratory, Boulder, CO

1. INTRODUCTION

Historically, the AWIPS site architecture has relied on a central repository at the site containing all data viewed at workstations within the site. The host containing this data is called the Data Server (DS). A file system containing the data is exported using NFS (Network File System) from the server to each of the visualization workstations at the site. This central Data Server has been one of the performance bottlenecks at the sites.

This architecture can be thought of as a *pull* approach, in which data is pulled by the workstation software as needed to service user−initiated display requests. FSL has investigated an alternative *push* approach, in which data is efficiently pushed to the workstations as it becomes available on the server. The data is then available on the local disk of the workstation, to be loaded as needed by the display software.

We expected this approach to improve workstations performance by reducing latencies and increasing throughput in displaying data on the workstations. As we show later, we have experimentally verified a performance improvement from this approach.

There are a number of open design issues, including the optimal subset of site data to broadcast, the data notification mechanisms, and whether to multicast transmission−format or decoded data formats. In our demonstration system, we've generally taken the simplest approach we could find in tackling each of these issues.

2. THE MULTICAST APPROACH

To demonstrate efficiently pushing data to the workstations, FSL has adopted the broadcast software developed at the National Severe Storms Laboratory (NSSL). This software was developed for use in radar product generation, but is sufficiently general to allow other uses such as ours. The NSSL software may be run as either a *broadcast*, in which data is sent to every host on a network, or as a *multicast,* in which data is sent to an addressable subset of the hosts on a network. We are using the software in the latter, multicast mode.

Broadcast (and multicast) sends a single stream of packets across the network, which are received simultaneously by all the receiving hosts. This obviously results in dramatically less network traffic than repetitively sending the same data over dedicated connections to each receiving host. Normally, broadcast and multicast are considered to be unreliable communication mechanisms, in the sense that data is not guaranteed to be delivered, or to arrive in the correct order. The NSSL software builds an additional level of reliability, by checking for data arrival, and retransmitting any missing data over a reliable channel. In practice we have found that very little data needs to be resent.

To reduce the amount of interprocess−communications (IPC) middleware software in our demonstration system, we replaced the NSSL IPC software (known as Linear Buffer) with the standard AWIPS IPC software (known as ThreadIPC).

The two primary NSSL processes are named bcast and brecv (for broadcast, and broadcast receiver, respectively). We modifed these processes slightly, and to distinguish the modified versions, we've renamed them to mcast and mrecv (for multicast and multicast receiver). There is one mcast process on the central server, and one mrecv process on every workstation host that receives the multicast data.

3. EXPECTED BENEFITS

Our rationale in implementing the AWIPS multicast system was to improve overall system performance and expandability. We expected a number of specific benefits.

3.1 Loading Time Improvements

Since the workstations can now load data directly from a local disk, rather than across the network from the NFS server, throughput should be higher, and latency lower, resulting in faster load times. This should be especially true for

* *Corresponding Author Address:* Michael R. Biere, NOAA/OAR/FSL, Systems Development Division, 325 Broadway R/FS4, Boulder, CO 80305−3328; e−mail: biere@fsl.noaa.gov.

[?] In collaboration with CIRA (Cooperative Institute for Research in the Atmospere), Colorado State University.

high−volume data sets such as satellite imagery.

3.2 Lowering NFS Server Load

Since the multicast data needs to be accessed only once to transmit, rather than for every workstation access, we expect the load on the data server disks to be reduced, along with the NFS load on the server.

3.3 Expandability

Adding additional workstations has no additional load to the server in our multicast architecture. We could also use this mechanism to provide data to local application servers with no impact on the AWIPS data server. (This is a slight simplification, but the additional overhead is very small compared with traditional access via the NFS server.)

3.4 Redundancy

Since we are broadcasting the data to every workstation, a server failure will not affect access to existing data. This is unlike the current architecture, in which loss of the data server is a critical failure requiring failover to an alternate server, during which time all data is unavailable.

3.5 Lower Network Usage

Along with lower server overhead from accessing data only once, the network usage is also lowered due to sending the data only once for all workstations.

3.6 No NFS Race Conditions

With the data securely on a local disk, there will be no nfs cache inconsistencies or race conditions such as those which plague AWIPS display software from time to time.

4. DESIGN ISSUES

In considering the use of a multicast data distribution at an AWIPS site, there are a number of design issues which need to be addressed. We list some of them here, and discuss the simplifying assumptions we made in our demonstration system.

4.1 What Data to Multicast

One extreme approach is to send and store all data locally on every workstation. In this case, there would be no need for a central data repository.

An alternative is to send all data, but keep a smaller subset of it on the workstations than on a central data server. The central server might have a longer history of data, or case studies, for example.

Another possibility is to only send a subset of the data to all workstations, and rely on a server for the rest. The criteria for determining what data to send might be based on timeliness performance requirements, or design expediency. For example, satellite and radar data files are good candidates because they are both important to the forecaster, and are file−based data sets and hence easy to broadcast, as we discuss next. Those are the data sets we chose for our demonstration.

4.2 Structured Data Sets

A few of the AWIPS data sets are stored in a simple format, with one data set per file. For example, radar scans and satellite images are stored in individual files. This makes distributing them easy, since as each file appears on the server, it is just copied to the workstations.

Many data sets are more complicated. For example, model runs are stored in structured files, but fields of each model run are received individually, and inserted into place in the structured file. If the individual fields are multicast as they are received, then the multicast receiver software must replicate the ingest functionality that decodes and stores the data into the appropriate file. The extreme case of this would be to broadcast all data as received, in transmission format, and replicate all of the decoders on each workstation.

Alternatively, one could wait until each model run (or other structured file) is finished, and then multicast the entire file. This would result in considerable delay in the availability of the earliest data within the structured file.

A compromise implementation would be for the workstation display software to use the existing server files while they are being updated, and use local multicast copies once the files are complete.

4.3 Inventory Server

The current AWIPS software is limited in its ability to accommodate an architecture wherein some data is stored on the server, and some on each workstation. In particular, AWIPS display software assumes that all meteorological data to

display resides within a single directory hierarchy pointed to by the FXA_DATA environment variable.

One can make some dramatic workarounds to this limitation by means of symbolic links within the directory structure, but a more elegant solution would be to extend the existing design with an inventory server software component. This inventory server would know about the actual distribution of data at the site, conceptually merging the central data with that at the workstations, and removing the limitation of one directory hierarchy.

4.3 Missed Data

What do we do when a workstation is down and unavailable for a while, then comes back up? One approach would be to initiate a backfill operation, filling in data from either a central server. Once the local data base was caught up, use of the workstation could resume.

Another approach is to merge any missing data into the local database as needed by the workstation software, in conjunction with loading it to the display.

The simplest and perhaps best approach is not to bother with backfilling at all, but merge in any missed data from the central server as needed while the workstation is running. This approach merges well with the inventory server previously discussed.

In any of these approaches, the role of central server could be played by another workstation (or even a set of workstations in Napster−like fashion) with a complete database.

4.4 Relational Database

Our description of data storage within AWIPS has been simplified to this point. In addition to the file−based data sets which we have been discussing, other data is currently stored in an Informix relational database on the server. It seems unreasonable to replicate this Informix database on every workstation, so we assume that these data sets will remain centralized.

4.5 Notification Server

The AWIPS notification server process is responsible for telling client processes at the site when new data is available for display. The notification server has been a centralized function at the site. Once data is distributed to individual workstations, each workstation has a potentially different view of the arrival of data. Hence in a robust architecture, each workstation should have its own notification server, reporting the arrival of data to the workstation. This is similar to the need for a distributed inventory server, reflecting the new distributed nature of data availability.

Use of the central notification server in the distributed data environment is subject to race conditions in which notifications arrive before data has been completely received by the multicast receiver on the workstation.

4.6 Purging

Although it's a relatively minor issue, it's clear that replicating data on the workstations will also require replicating data management activities such as purging old data sets to make room for new ones.

5. DEMONSTRATION SYSTEM

FSL implemented a demonstration multicast system for the two simplest (but still important!) data sets: satellite images and radar tilts. We have taken the path of simplicity in most of the design decisions. Figure 1 shows the key processes in our system. The left side shows the processes on the data server, while the right side shows the processes that are replicated on every workstation. The dashed line down the center represents the local area network.

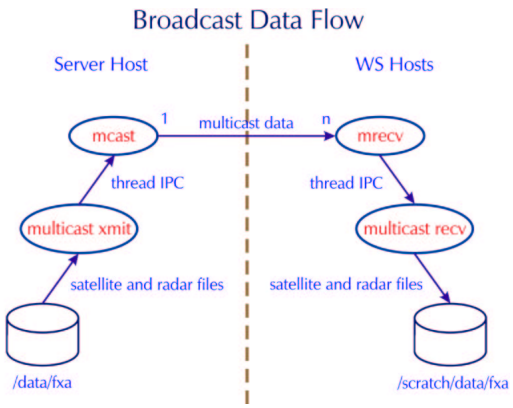

Figure 1. Process data flow of demonstration multicast system.

The mcast and mrecv processes are modified versions of the NSSL bcast and brecv processes. The multicast_xmit and multicast_recv processes are client processes which handle reading and writing of AWIPS satellite and radar files.

On the server, the multicast_xmit process registers with the notification server to be notified

when new radar or satellite data are available. When a new file is available, it is read from the data tree below /data/fxa in the file system. The data is then sent via the AWIPS thread IPC to the mcast process, which multicasts it to the workstations.

On each workstation, the mrecv process receives the multicast data and hands it off to the client process multicast_recv, which writes the file to disk. The data is written to the workstation local disk in a different directory hierarchy, for example at /scratch/data/fxa. Every file is written to the same relative position in this file hierarchy as where it was read from in the server's /data/fxa hierarchy.

6. EXPERIMENTAL RESULTS

The test environment simulated the most current AWIPS−deployed system including the linux preprocessors and linux workstation. All workstation tests were performed on linux systems.

The system had three areas to test:

1) performance for the forecaster;
2) impact on the data server; and
3) data availability and validity.

6.1 Performance Tests

The performance tests run three scripting routines that simulate a forecaster using a mouse and making menu selections. The three tests load satellite, radar, and synoptic data sets. The time lapse for running the tests is then averaged and a rating is determined from that average.

Figure 2 shows a high level system and network configuration used for these tests. The hybrid computer architecture uses an HP NFS server and linux NFS server. The HP system is networked by FDDI to a Plaintree switch. The data is then transmitted over a 100baseT network to the forecaster workstation. During the next year, the NWS will beta test this system with the satellite and grid data available on the linux server and other data sets on the HP server.

Four timing tests were run on different configurations for accessing data. The linux workstation was staged with AWIPS 5.2.2 software and care was taken during the test to remove all of the buffered cache data in memory. Figure 3 shows the configuration for the first test.

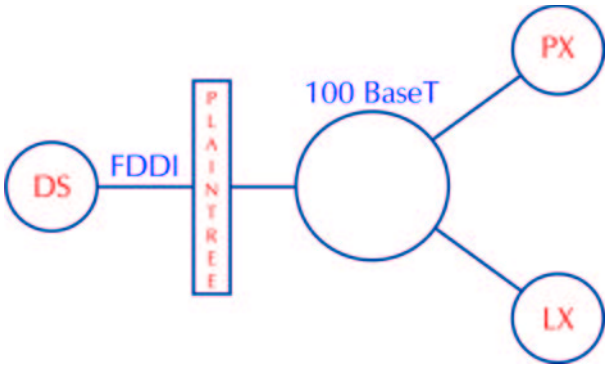

Figure 2. Test system configuration.

For general information, a test was run with all of the test data moved to a RAM filesystem on the workstation. Although this would not represent a operational system, this gives a good benchmark for the performance tests. Since the data is resident in memory, no disk or network interference is possible. The results represent the best that the workstation can perform on the benchmark tests. Figure 3 shows the system configuration.

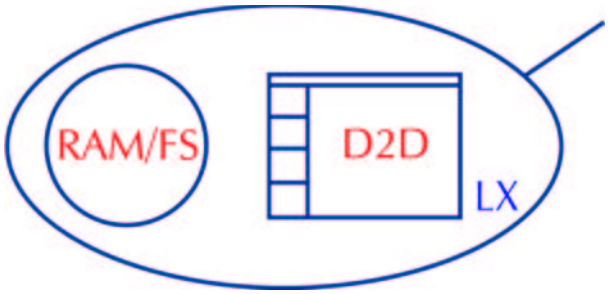

Figure 3. RAM filesystem configuration.

For the second performance test, all of the data was moved locally to a disk on the forecaster workstation. This test shows the performance benchmarks when an NFS server is not used to host data.

The third performance put the data repository on the linux fileserver (see Figure 4). The workstation accessed data via NFS over the 100BaseT network to transfer the frames for display. This is similar to current schemes that the NWS is investigating with port to linux.

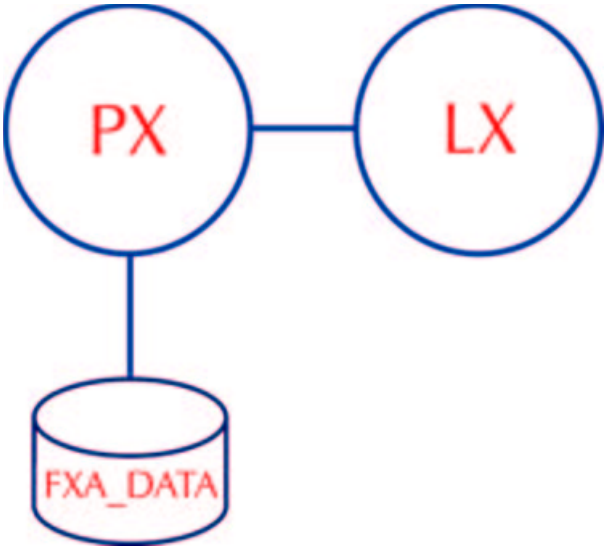

Figure 4. Linux NFS data server.

The fourth configuration tested had the data resident on the HP data NFS server. All data was transferred to the linux workstation via the FDDI and network switch. Figure 2 reflects this configuration.

The final tests reflect the b−cast data repository. The grids are on the linux preprocessor, the point data is on the HP data server, and the radar and satellite data are local to the workstation (see Figure 5).

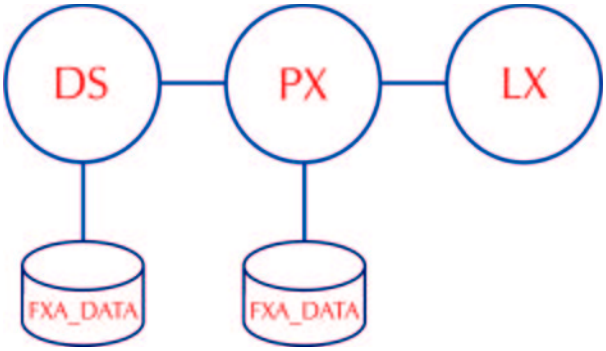

Figure 5. Simulated multi−cast configuration.

The performance benchmark is based on software written by Michael Biere and Timothy Hopkins (National Weather Service). These scripts characterize data retrieval for normal use of the workstation by a forecaster. The final performance ratings from the aforementioned tests are summarized in Table 1.

The last line in Table 1 shows the WPR for the HP workstation client and server. These ratings reflect the current AWIPS system minus any linux upgrades. As one can see, the performance improvements are remarkable the more that data is removed from the HP NFS server.

File access type	*WPR*
RAM filesystem	28
Local data	35
Linux NFS	59
HP NFS	95
B−cast (Figure 5.)	44
All HP hardware	>300

Table 1. WPR ratings for testing configurations.

6.2 Impact and Data Tests

The tests for server impact and data validity have not been completed at this time. These results will be presented with the paper at the conference.

7. SUMMARY

In summary, although the software is in preliminary testing, the results are promising. The forecasters will see performance gains for each data set propagated to the linux server and workstation. As expected, the greatest gains show all the data local to the forecaster workstation. This, however, is a complicated data management effort. Separating and prioritizing which data sets is key to a successful use of the b−cast software for system support and forecasters alike.

8. REFERENCES

Jing, Z., M. Jain, 2000: The Linear Buffer and its Role in the WSR−88D Open System RPG, *Preprints, 16th International Conference IIPS*, Long Beach, CA, Amer. Meteor. Soc., Paper 11.9, pp 395−398.

Jain, M., Z. Jing, A. Zaharai, A. Dodson, H. Burchan, D. Priegnitz, S. Smith, 1997: Software Architecture of the NEXRAD Open Systems Radar Product Generator (ORPG), *Preprints 13th International Conference IIPS*, Long Beach, CA, Amer. Meteor. Soc., Paper 8.4, pp 238−241.